Introductory Numerical Analysis

Thomas Trogdon
University of Washington
trogdon@uw.edu

Contents

Preface

My preface

Chapter 0

Matlab basics

MATLAB is a scripting language without types. That means a few things.

1. There is no need to declare variables before you assign values to them. This, for example, is something that is necessary in JAVA.

2. A variable initially defined to be a scalar than then be assigned to be an array and then assigned to be a function. This can be convenient but also makes it tougher to catch bugs.

3. Unless a specific environment is loaded, MATLAB will do all of its calculations in floating point arithmetic. In a language like PYTHON an expression like `1/4` will return `0` because `0` is the closest integer and `1` and `4` are integers. MATLAB will return `0.25`.

MATLAB code is stored in `.m`-files, or m-files for short. These are also referred to as MATLAB scripts. It is good practice to include the following at the top of every script that your write:

```
clear all;  close all;
```

This will help ensure that

1. The ability of your code to run is not affected by other previously set variables.

2. If you close MATLAB, your script will behave the same when the next time you open it.

MATLAB will also print the output of any line if you do not append a semicolon `;` at the end of the line. For example, executing the following produces output to the Command Window

```
x = 10
```

```
x =

    10
```

while

```
x = 10;
```

produces no output. In order to debug your code you will want to have nearly every line end with a semicolon. And then if the script does not run as expected you can remove one or two semicolons at a time to monitor the output.

The following code uses a *for loop* to add up all the positive integers that are less than or equal to n

```
n = 10;
SUM = 0; % using capital letters because sum() is a built-in function
for i = 1:n
    SUM = SUM + i;
end
SUM
n*(n+1)/2  % known answer to check
```

```
SUM =

    55


ans =

    55
```

Another type of loop is the *while loop*. Here is an example of performing the same sum as above using a while loop.

```
n = 10;
SUM = 0;
i = 0;
while i < n
    i = i + 1;
    SUM = SUM + i;
end
SUM
```

```
SUM =

    55
```

0.1 ▪ Using Matlab with some examples from calculus

Because Part I of this text concerns numerical linear algebra, there are few opportunities introduce the reader to the plotting functionality in MATLAB. So, we take some time to review some theorems from calculus and illustrate them using MATLAB.

0.1.1 ▪ Demonstrations from differential calculus

Definition 0.1. *A function* f *defined on a set* X *has limit* L *at* x_0

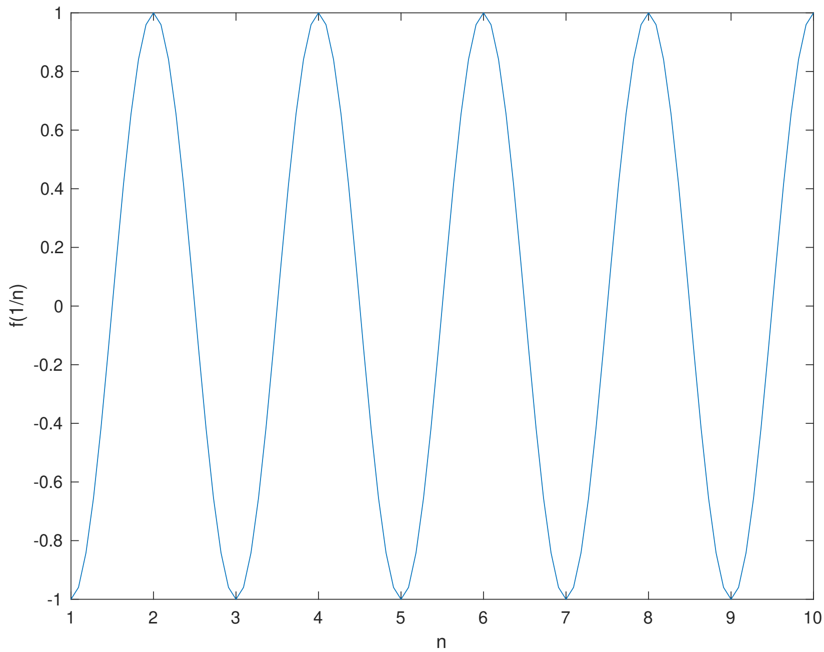

Figure 1. *A plot of the function $f(1/n)$*

$$\lim_{x \to x_0} f(x) = L$$

if, given any real number $\epsilon > 0$, there exists a real number $\delta > 0$ such that

$$|f(x) - L| < \epsilon, \quad \textit{whenever} \quad x \in X \quad \textit{and} \quad 0 < |x - x_0| < \delta.$$

The function

$$f(x) = \begin{cases} \cos(\pi/x), & x \neq 0, \\ 1 & x = 0, \end{cases}$$

is not continuous at $x = 0$. To show this, let $x_n = 1/n$. If f is continuous then $\lim_{n\to\infty} f(1/n) = 1$

```
f = @(x) cos(pi./x);
ns = linspace(1,10,100);
plot(ns,f(1./ns))
xlabel('n'); ylabel('f(1/n)') %label axes
```

The plot that results is shown in Figure 1

Definition 0.2. *Let $C^n[a, b] = \{f : [a, b] \to \mathbb{C} \mid f^{(n)}$ exists and is continuous$\}$.*

We use the notation $C[a, b] := C^0[a, b]$.

Theorem 0.3 (Intermediate Value Theorem). *Let $f \in C[a, b]$. Assume $f(a) \neq f(b)$.*

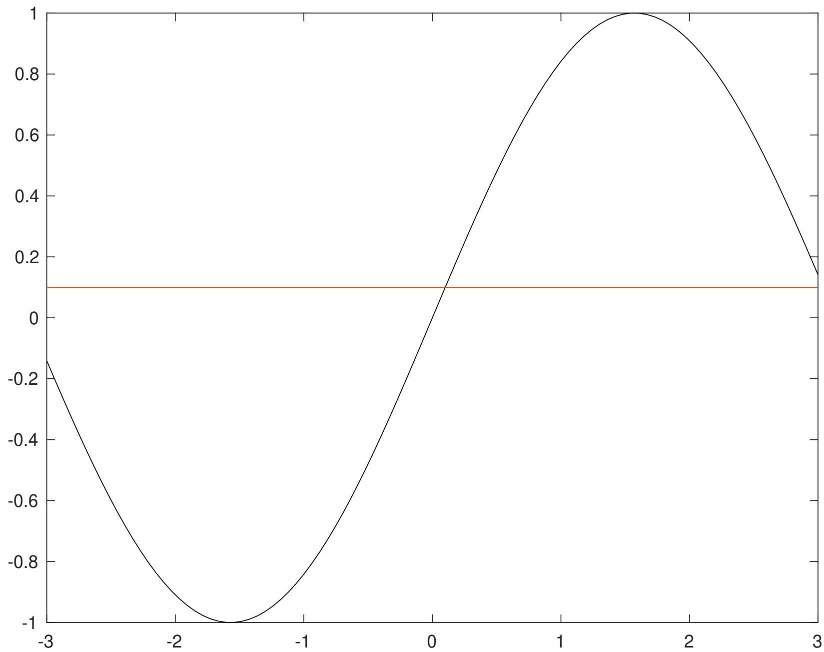

Figure 2. *A demonstration of the intermediate value theorem for* $f(x) = \sin(x)$.

For every real number y, $f(a) \leq y \leq f(b)$, *there exists* $c \in [a, b]$ *such that* $f(c) = y$.

```
x = linspace(-3,3,100);
f = @(x) sin(x);
c = @(x) 0*x+.1;
plot(x,f(x),'k')
hold on
plot(x,c(x)) %every value between f(-3) and f(3) is attained at least once
```

The plot that results is shown in Figure 2

Definition 0.4. *Let* f *be a function defined on an open interval containing* x_0. *The function* f *is differentiable at* x_0 *if*

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

exists. In which case, $f'(x_0)$ *is the derivative of* $f(x)$ *at* x_0. *If* f *has a derivative at each point in a set* X *then* f *is said to be differentiable on* X.

```
format long %to see more digits
f = @(x) sin(x); df = @(x) cos(x);
x = .0001; x0 = 0;
(sin(x)-sin(x0))/(x-x0)-cos(x0)
```

```
ans =
```

```
-1.666666582522680e-09
```

Here are some of the most important theorems from single-variable calculus:

Theorem 0.5. *If a function f is differentiable at x_0, it is continuous at x_0.*

Theorem 0.6 (Rolle's Theorem). *Suppose $f \in C[a,b]$ and f is differentiable on $[a,b]$. If $f(a) = f(b)$, the a number c in (a,b) exists with $f'(c) = 0$.*

Theorem 0.7 (Mean Value Theorem). *Suppose $f \in C[a,b]$ and f is differentiable on $[a,b]$. There exists a point $c \in (a,b)$ such that*

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

Theorem 0.8 (Extreme Value Theorem). *If $f \in C[a,b]$, then $c_1, c_2 \in [a,b]$ exist with $f(c_1) \leq f(x) \leq f(c_2)$, for all $x \in [a,b]$. Furthermore, if f is differentiable on $[a,b]$ then c_1, c_2 are either the endpoints (a or b) or at a point where $f'(x) = 0$.*

This theorem states that both the maximum and minimum values of $f(x)$ on a closed interval $[a,b]$ must be attained within the interval (at points c_2 and c_1). A more involved theorem is the following:

Theorem 0.9. *Suppose $f \in C[a,b]$ is n-times differentiable on (a,b). If $f(x) = 0$ at $n+1$ distinct numbers $a \leq x_0 < x_1 < \cdots < x_n \leq b$, then a number $c \in (x_0, x_n)$, (and hence in (a,b)) exists with $f^{(n)}(c) = 0$.*

Consider the fourth degree polynomial $f(x) = 8x^4 - 8x^2 + 1$:

```
f = @(x) 8*x.^4-8*x.^2+1; % has 4 zeros on (-1,1)
dddf = @(x) 8*4*3*2*x; % must have 1 zero on (-1,1)
x = linspace(-1,1,100);
plot(x,dddf(x))
```

The plot that results is shown in Figure 3

Theorem 0.10 (Taylor's Theorem). *Suppose $f \in C^n[a,b]$, and that $f^{(n+1)}$ exists on $[a,b]$, and $x_0 \in [a,b]$. For every $x \in [a,b]$, there exists a number $\xi(x)$ between x_0 and x with*

$$f(x) = P_n(x) + R_n(x),$$

where

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n,$$

and

$$R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{n+1}.$$

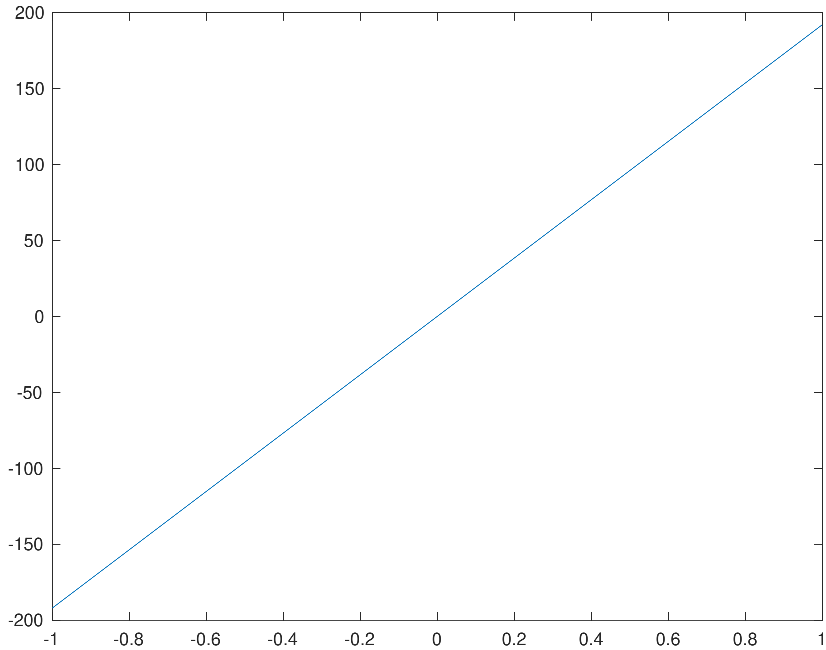

Figure 3. *A demonstration of Theorem 0.9 with* $f(x) = 8x^4 - 8x^2 + 1$.

0.1.2 ▪ Demonstrations from integral calculus

Definition 0.11. *The Riemann integral of the function f defined on the interval $[a, b]$ is the following limit (if it exists):*

$$\int_a^b f(x)dx = \lim_{\max \Delta x_i \to 0} \sum_{i=1}^{n} f(\bar{x}_i)\Delta x_i,$$

where the numbers $x_0, x_1, \ldots, x_n$ satisfy $a = x_0 \leq x_1 \leq \cdots \leq x_n = b$, $\Delta x_i = x_i - x_{i-1}$ for $i = 1, 2, \ldots, n$. And $\bar{x}_i$ is an arbitrary point in the interval $[x_{i-1}, x_i]$.

Let's choose the points x_i to be evenly spaced: $x_i = a + i\frac{b-a}{n}$ and $\bar{x}_i = x_i$. Then we have $\Delta x_i = \frac{b-a}{n}$ and

$$\int_a^b f(x)dx = \lim_{n\to\infty} \frac{b-a}{n} \sum_{i=1}^{n} f(x_i).$$

```
f = @(x) exp(x);
n = 10; a = -1; b = 1;
x = linspace(a,b,n+1); % create n + 1 points
x = x(2:end); % take the last n of these points
est = (b-a)/n*sum(f(x)) % evaluate f at these points and add them up
actual = exp(b)-exp(a) % the actual value
abs(est-actual)
```

```
est =
```

```
    2.593272082493666


actual =

    2.350402387287603


ans =

    0.242869695206063
```

Now choose and $\bar{x}_i = \dfrac{x_i + x_{i-1}}{2}$ to be the midpoint. We still have $\Delta x_i = \frac{b-a}{n}$ and

$$\int_a^b f(x)dx = \lim_{n\to\infty} \frac{b-a}{n} \sum_{i=1}^{n} f(\bar{x}_i).$$

```
f = @(x) exp(x);
n = 10; a = -1; b = 1;
x = linspace(a,b,n+1); % create n + 1 points
x = x(2:end); % take the last n of these points
x = x - (b-a)/(2*n); % shift to the midpoint
est = (b-a)/n*sum(f(x)) % evaluate f at these points and add them up
actual = exp(b)-exp(a) % the actual value
abs(est-actual)
```

```
est =

    2.346489615388305


actual =

    2.350402387287603


ans =

    0.003912771899298
```

Theorem 0.12 (Weighted Mean Value Theorem). *Suppose $f \in C[a,b]$, the Riemann integral of g exists on $[a,b]$, and $g(x)$ does not change sign on $[a,b]$. Then there exists a number c in (a,b) with*

$$\int_a^b f(x)g(x)dx = f(c)\int_a^b g(x)dx.$$

Part I

Numerical linear algebra

Chapter 1

Systems of equations

Before we start combining mathematics with programming in MATLAB we need to learn some of the mathematics!

1.1 ▪ A linear system of equations

Consider the following basic problem of fitting a line to data:

Problem 1.1. *Find the line that passes through the coordinates* $(0, 1)$ *and* $(-1, 2)$.

To solve this problem we start with our general form for a line

$$y = f(x) = ax + b,$$

and then enforce the conditions $f(0) = 1$ and $f(-1) = 2$. This gives

$$\begin{aligned} a \cdot 0 + b &= 1, \\ a \cdot (-1) + b &= 2. \end{aligned}$$

This is a *linear system* of two equations for the two unknowns a, b. Below we will make the term linear precise and we will understand when we can solve such a system. We will also extend our understanding to systems of n equations for n unknowns. And even go further and generalize this to systems m linear equations with n unknowns!

Back to the problem at hand, we see that the first equation tells us $b = 1$ and the second equation gives $-a + 1 = 2$, or $a = -1$. So the line

$$y = -x + 1,$$

passes through the coordinates $(0, 1)$ and $(-1, 2)$.

1.1.1 ▪ Vectors

A vector can be understood as an $n \times 1$ array of numbers:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}.$$

We will also use the shortened notation $\mathbf{v} = (v_j)_{j=1}^n$ to represent the same vector. We refer to the v_j's as the entries, or components, of $\mathbf{v}$. Unless otherwise stated, for a vector $\mathbf{v}$, v_j will refer to its entries. Sometimes we will encounter *row vectors.* A row vector is a $1 \times n$ array of numbers.

Some more notation before we proceed:

- $\mathbb{R}$ and $\mathbb{C}$ refer to all real and complex numbers[1], respectively.
- To state that a is a real number, we write $a \in \mathbb{R}$ (and similarly $a \in \mathbb{C}$) if a is complex).
- $\mathbf{v}$ is a vector with n entries, all being real numbers, we say $v \in \mathbb{R}^n$ (and similarly $\mathbf{v} \in \mathbb{C}^n$) if $\mathbf{v}$ has complex entries).
- We use the notation $\sum_{j=1}^n v_j = v_1 + v_2 + \cdots + v_n$.

Vectors represent an extension of our usual real (or complex) number system and so we should give some rules for addition, subtraction, and multiplication.

Definition 1.2. *We use the following rules for manipulating vectors $\mathbf{u} = (u_j)_{j=1}^n, \mathbf{v} = (v_j)_{j=1}^n$ in $\mathbb{R}^n$:*

1. $\mathbf{u} + \mathbf{v} = (u_j + v_j)_{j=1}^n$,
2. $\mathbf{u} - \mathbf{v} = (u_j - v_j)_{j=1}^n$,
3. $a\mathbf{v} = (av_j)_{j=1}^n$ *for* $a \in \mathbb{R}$, *and*
4. $\mathbf{u} = \mathbf{v}$ *if and only if* $\mathbf{u} - \mathbf{v} = \mathbf{0}$ *where* $\mathbf{0} \in \mathbb{R}^n$ *is the vector of all zeros.*

These rules state that everything just happens entry-by-entry. The last statement is equivalent to $u_j = v_j$ for every j. Note that enforcing that a vector with n entries is zero is actually enforcing n conditions! We also see the convenience of the notation introduced above.

Example 1.3. Let $\mathbf{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{w} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Then

- $\mathbf{v} + \mathbf{w} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$, and
- $2\mathbf{w} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$.

Then Problem 1.1, a system of two equations, can be summarized using one *vector equation.* To see how to do this first note that

$$\begin{bmatrix} a \cdot 0 \\ a \cdot (-1) \end{bmatrix} = a \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} b \\ b \end{bmatrix} = b \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

So then we need to enforce that

$$a \begin{bmatrix} 0 \\ -1 \end{bmatrix} + b \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

This conversion of the problem does not actually help us solve it! We need to introduce matrices before this conversion really becomes useful.

[1] A complex number z is equal to $z = x + \mathrm{i}\, y$ where $x, y \in \mathbb{R}$ and $\mathrm{i} = \sqrt{-1}$. While these numbers may feel strange and artificial, they are extremely helpful in simplifying computations, especially on a computer.

1.1.2 ▪ Matrices

A matrix can be understood as an $m \times n$ array of numbers

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & \vdots \\ \vdots & & \ddots & \\ a_{m1} & \cdots & & a_{mn} \end{bmatrix}.$$

And just like the case of vectors we use the notation $A = (a_{ij})_{\substack{1\le i\le m \\ 1\le j\le n}}$ to quickly refer to the matrix. In this index notation we always take the top index (i in this case) to refer to the rows of the matrix and the bottom index (j) to refer to columns. Unless otherwise stated, for a matrix A, a_{ij} will refer to its entries.

Example 1.4. Write down the matrix A given by

$$A = \left(\frac{1}{i+j}\right)_{\substack{1\le i\le 3 \\ 1\le j\le 4}}.$$

The matrix A is then given by

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}.$$

Similar to vectors, if A is an $m \times n$ matrix of real numbers, we write $A \in \mathbb{R}^{m\times n}$ (similarly $A \in \mathbb{C}^{m\times n}$ if A has complex entries). And we have an analogous set of rules.

Definition 1.5. *We use the following rules for manipulating matrices* $A = (a_{ij})_{\substack{1\le i\le m \\ 1\le j\le n}}, B = (b_{ij})_{\substack{1\le i\le n \\ 1\le j\le m}}$ *in* $\mathbb{R}^{m\times n}$*:*

1. $A + B = (a_{ij} + b_{ij})_{\substack{1\le i\le m \\ 1\le j\le n}}$,
2. $A - B = (a_{ij} - b_{ij})_{\substack{1\le i\le m \\ 1\le j\le n}}$,
3. $cA = (ca_{ij})_{\substack{1\le i\le m \\ 1\le j\le n}}$ *for* $c \in \mathbb{R}$, *and*
4. $A = B$ *if and only if* $A - B = \mathbf{0}$ *where* $\mathbf{0} \in \mathbb{R}^{m\times n}$ *is the matrix of all zeros.*

As with vectors, addition, multiplication and subtraction are simply taken entry-by-entry. Note that matrix sizes must be the same for them to added or subtracted.

Example 1.6. Let $A = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & -1 \end{bmatrix}$. Then

$$A + B = \begin{bmatrix} 2 & 2 & 0 \\ -2 & 0 & 0 \end{bmatrix},$$

$$3A = \begin{bmatrix} 3 & 6 & 0 \\ -3 & 0 & 3 \end{bmatrix}.$$

Matrix-vector multiplication

We now return to this equation

$$a\begin{bmatrix}0\\-1\end{bmatrix}+b\begin{bmatrix}1\\1\end{bmatrix}=\begin{bmatrix}0&1\\-1&1\end{bmatrix}\begin{bmatrix}a\\b\end{bmatrix}=\begin{bmatrix}1\\2\end{bmatrix}.$$

The first equality can be taken as a definition of the matrix-vector product and it motivates the general definition below. But before we discuss this in greater detail we introduce some notation for columns of a matrix. Suppose $A \in \mathbb{R}^{m\times n}$. The vectors $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n \in \mathbb{R}^m$ are the columns of A, i.e.,

$$A=\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n\end{bmatrix}.$$

Definition 1.7. *Suppose $A \in \mathbb{R}^{m\times n}$ and $\mathbf{v} \in \mathbb{R}^n$ then we define*

$$A\mathbf{v}=\sum_{j=1}^{n} v_j \mathbf{a}_j.$$

In other words, the matrix-vector product $A\mathbf{v}$ gives a weighted sum of the columns of A.

Example 1.8.

$$\begin{bmatrix}0&1\\-1&1\end{bmatrix}\begin{bmatrix}-1\\1\end{bmatrix}=(-1)\begin{bmatrix}0\\-1\end{bmatrix}+(1)\begin{bmatrix}1\\1\end{bmatrix}=\begin{bmatrix}1\\2\end{bmatrix}.$$

But this is not typically how one computes this by hand. By hand, most students find it easier to use the rule

Theorem 1.9. *For $A \in \mathbb{R}^{m\times n}$ and $v \in \mathbb{R}^n$*

$$A\mathbf{v}=\left(\sum_{j=1}^{n} a_{ij}v_j\right)_{i=1}^{m}.$$

Example 1.10. To compute

$$\mathbf{w}=\underbrace{\begin{bmatrix}1&0&-1\\1&2&3\end{bmatrix}}_{2\times 3}\underbrace{\begin{bmatrix}1\\0\\-1\end{bmatrix}}_{3\times 1}$$

we first note that the "inner dimensions" match and therefore we have a well-defined product. The dimension of the output is calculated by

$$(2\times 3)\times(3\times 1)\longrightarrow(2\times \cancel{3})\times(\cancel{3}\times 1)\longrightarrow 2\times 1.$$

So, $\mathbf{w} \in \mathbb{R}^2$. By Theorem 1.9 we see that

$$w_1 = 1 \cdot 1 + 0 \cdot 0 + (-1) \cdot (-1) = 2.$$

This is computed by multiplying the first row of the matrix times the vector and adding. Similarly, the second entry is given by

$$w_2 = 1 \cdot 1 + 2 \cdot 0 + 3 \cdot (-1) = -2.$$

We conclude that $\mathbf{w} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$. Let's now go back and verify that this gives the same as our definition:

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = 1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0 \begin{bmatrix} 0 \\ 2 \end{bmatrix} + (-1) \begin{bmatrix} -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}.$$

Computing a matrix-vector product via Theorem 1.9 is simpler by hand because you can work entry-by-entry and you do not have to think about all of the columns at once. But for a computer, thinking about all the columns is actually a simpler way of doing things! This highlights something interesting that we'll encounter again, many times, in this book:

- Something that is hard for human computation may be "easy" for a computer.

1.1.3 ▪ Matrix-matrix product

While it is not immediately clear why we will ever need to develop a way to multiply two matrices, we define it here. It turns out this is absolutely critical!

Definition 1.11. *Suppose $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{\ell \times m}$. Then*

$$BA = \begin{bmatrix} B\mathbf{a}_1 & B\mathbf{a}_2 & \cdots & B\mathbf{a}_n \end{bmatrix}.$$

Note that

$$\underbrace{B}_{\ell \times m}\underbrace{A}_{m \times n},$$

so the output should be

$$(\ell \times m) \times (m \times n) \longrightarrow (\ell \times \cancel{m}) \times (\cancel{m} \times n) \longrightarrow \ell \times n.$$

This is confirmed by noting that $B\mathbf{a}_j \in \mathbb{R}^\ell$ and we have n of these vectors. To actually compute the matrix-matrix product by hand, it is often easiest to fill out each column of the resulting matrix at a time, by doing a matrix-vector product.

Example 1.12.

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ -2 & 6 \end{bmatrix}$$

1.2 ▪ Creating and using vectors and matrices in Matlab

This section is the first example of what will become common in this text. Right after some new mathematical concepts are introduce, we realize them using MATLAB code.

1.2.1 ▪ Vectors

Here is the creation of a row vector `v` and a (column) vector `w`.

```
v = [1,2,3]; % row vector
w = [1;2;3]; % column vector
v
w
```

```
v =

     1     2     3


w =

     1
     2
     3
```

Now, let us create these vectors and then grab specific entries or groups of entries. The last command here add up all the entries in a vector. The output that follows gives the result from lines 3-6.

```
1 v = [1,2,3]; % row vector
2 w = [1;2;3]; % column vector
3 w(1)
4 v(1:2)
5 w(2:3)
6 sum(w)
```

```
ans =

     1


ans =

     1     2


ans =

     2
     3
```

```
ans =

     6
```

1.2.2 ▪ Matrices

Now, we will create a matrix by manually specifying every entry. MATLAB uses the semicolon ; as a line break.

```
M = [1,2,3,4;5,6,7,8;9,10,11,12;12,14,15,16]
M(1,4) % get the (1,4) element
M(2:3,2:3)% get the "middle" block of the matrix
M(3,:) % Get row 3
sum(M)
```

```
M =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    12    14    15    16


ans =

     4


ans =

     6     7
    10    11


ans =

     9    10    11    12


ans =

    27    32    36    40
```

For the last example in this section we confirm the multiplication in Example 1.12.

```
A = [1,0,-1; 1,2,3];
B = [1,1; 0,1; -1,1];
A*B
```

```
ans =

     2     0
    -2     6
```

1.3 ▪ Regular Gaussian elimination

Before we begin our discussion of Gaussian elimination we define some important concepts

Definition 1.13.

- *The diagonal of a matrix $A \in \mathbb{R}^{m\times n}$ is the entries a_{ii} for $i = 1, 2, \ldots, \min\{m, n\}$.*
- *A matrix L is said to be* lower triangular *if $\ell_{ij} = 0$ for all $i < j$.*
- *A matrix U is said to be* upper triangular *if $u_{ij} = 0$ for all $i > j$.*
- *An upper/lower-triangular matrix is said to be* special *if all the diagonal entries are ones.*

We use an example to illustrate (regular) Gaussian elimination. Consider the linear system of equations

$$\begin{aligned} x + 4y + z &= 2,\\ 2x + 12y + z &= 7,\\ x + 2y + 4z &= 3. \end{aligned} \tag{1.1}$$

We know that this system is equivalent to the vector equation

$$\begin{bmatrix} 1 & 4 & 1\\ 2 & 12 & 1\\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} x\\ y\\ z \end{bmatrix} = \begin{bmatrix} 2\\ 7\\ 3 \end{bmatrix}.$$

This last system is represented in shorthand by the *augmented matrix*

$$\left[\begin{array}{ccc|c} 1 & 4 & 1 & 2\\ 2 & 12 & 1 & 7\\ 1 & 2 & 4 & 2 \end{array}\right].$$

$$\begin{aligned} x + 4y + z &= 2\\ 2x + 12y + z &= 7\\ x + 2y + 4z &= 3 \end{aligned} \qquad \Leftrightarrow \qquad \left[\begin{array}{ccc|c} 1 & 4 & 1 & 2\\ 2 & 12 & 1 & 7\\ 1 & 2 & 4 & 2 \end{array}\right]$$

Now, on the left, we multiply the first equation by -2 and add it to the second equation

$$\begin{array}{rr} & (-2)\cdot(x + 4y + z = 2)\\ + & 2x + 12y + z = 7\\ \hline & 0x + 4y - z = 3 \end{array}$$

We write the system out and convert it to our augmented matrix

$$\begin{aligned} x + 4y + z &= 2 \\ 0x + 4y - z &= 3 \\ x + 2y + 4z &= 3 \end{aligned} \qquad \Leftrightarrow \qquad \left[\begin{array}{ccc|c} 1 & 4 & 1 & 2 \\ 0 & 4 & -1 & 3 \\ 1 & 2 & 4 & 2 \end{array}\right]$$

Then, the important observation is that the new augmented matrix can be found from the old by taking $(-2)\cdot$ (row one) + (row two) and replacing row two with this: $-2R_1 + R_2 \to R_2$. We call this an *elementary row operation.* Now, we eliminate x from the last equation:

$$\begin{array}{rr} & (-1)\cdot(x + 4y + z = 2) \\ + & x + 2y + 4z = 3 \\ \hline & 0x - 2y + 3z = 1 \end{array}$$

$$\begin{aligned} x + 4y + z &= 2 \\ 0x + 4y - z &= 3 \\ 0x - 2y + 3z &= 1 \end{aligned} \qquad \Leftrightarrow \qquad \left[\begin{array}{ccc|c} 1 & 4 & 1 & 2 \\ 0 & 4 & -1 & 3 \\ 0 & -2 & 3 & 1 \end{array}\right]$$

The last step of (regular) Gaussian elimination for this system is to eliminate y from the last equation

$$\begin{array}{rr} & (1/2)\cdot(4y - z = 3) \\ + & -2y + 3z = 1 \\ \hline & 0y + 5/2z = 5/2 \end{array}$$

$$\begin{aligned} x + 4y + z &= 2 \\ 0x + 4y - z &= 3 \\ 0x + 0y + \frac{5}{2}z &= \frac{5}{2} \end{aligned} \qquad \Leftrightarrow \qquad \left[\begin{array}{ccc|c} 1 & 4 & 1 & 2 \\ 0 & 4 & -1 & 3 \\ 0 & 0 & \frac{5}{2} & \frac{5}{2} \end{array}\right]$$

This is the last step of Gaussian elimination.

Definition 1.14. *An elementary row operation of type I is of the form $aR_j + R_i \to R_i$ for $i \neq j$.*

Definition 1.15. *Regular Gaussian elimination is the process by which a matrix (square or rectangular) is reduced to upper triangular form using only row operations of type I where $i > j$.*

This definition states that in regular Gaussian elimination we can use only rows above to introduce zeros.

Algorithm 1: Regular Gaussian elimination `rGE`

```
Result: An upper triangular matrix, or failure
Input: A ∈ ℝ^{m×n}
begin
    for j = 1 to n do
        if if a_jj = 0 then
            A is not regular;
            return failure;
        else
            for i = j + 1 to n do
                set l_ij = a_ij / a_jj;
                perform −ℓ_ij R_j + R_i → R_i on A;
            end
        end
    end
end
```

We now need to understand why introducing zeros in the matrix, i.e. elimination, is beneficial. For the system

$$\begin{aligned} x + 4y + z &= 2 \\ 0x + 4y - z &= 3 \\ 0x + 0y + \frac{5}{2}z &= \frac{5}{2} \end{aligned}$$

we know that the solutions of this are the same as the solutions of the original system (1.1). We first determine z:

$$\frac{5}{2}z = \frac{5}{2} \Rightarrow z = 1,$$

then determine y

$$4y - z = 3 \Rightarrow y = 1,$$

and, lastly, determine x

$$x + 4y + z = 2 \Rightarrow x = -3.$$

This process of working backward up the matrix is referred to as *back substitution*. To properly describe this process algorithmically, consider breaking up the final augmented matrix

$$\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 4 & 1 & 2 \\ 0 & 4 & -1 & 3 \\ 0 & 0 & \frac{5}{2} & \frac{5}{2} \end{array}\right].$$

Then the following algorithm is back substitution.

Algorithm 2: Back substitution `Backsub`

Result: The solution of $U\mathbf{x} = \mathbf{c}$
Input: An augmented matrix $\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right]$, $U \in \mathbb{R}^{n\times n}$ upper triangular and $u_{jj} \neq 0$ for all j
begin
  set $x_n = c_n/u_{nn}$;
  for $i = n-1$ *to* 1 *with increment* -1 **do**
    set $x_i = \frac{1}{u_{ii}}\left(c_i - \sum_{j=i+1}^{n} u_{ij}x_j\right)$;
  end
end

Remark 1.16. *We return to the example linear system from Problem 1.1, now written in augmented form,*

$$\left[\begin{array}{cc|c} 0 & 1 & 1 \\ -1 & 1 & 2 \end{array}\right].$$

It is clear that regular Gaussian elimination will immediately fail on this matrix. We will introduce more row elementary row operations soon to deal with this system.

So, we now give a name to matrices for which regular Gaussian elimination succeeds.

Definition 1.17. *A matrix A is called regular if regular Gaussian elimination succeeds in transforming it to an upper-triangular matrix with non-zero diagonal entries.*

1.4 ▪ The LU factorization

Now consider the matrix product (by divine inspiration!)

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & \frac{5}{2} \end{bmatrix} = \begin{bmatrix} 1 & 4 & 1 \\ 2 & 12 & 1 \\ 1 & 2 & 4 \end{bmatrix} = A.$$

This is called an LU factorization of the matrix A. Note that the first matrix in the product is lower triangular with ones on the diagonal. How is this useful? To really understand this we need to discuss matrix inverses.

Definition 1.18 (Matrix inverse). *The inverse of an $n\times n$ square matrix A is another matrix B such that*

$$BA = I = I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}.$$

We refer to $I = I_n$ as the $n \times n$ identity matrix and we denote $B = A^{-1}$.

Example 1.19. For a 1×1 matrix $A = [a]$, where $a \in \mathbb{R}$, it is clear that $A^{-1} = [1/a]$.

But note that for

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

it is no longer clear what A^{-1} is!

Multiplication by the identity matrix does not change the vector.

Proposition 1.20. *For* $\mathbf{v} \in \mathbb{R}^n$, $I_n \mathbf{v} = \mathbf{v}$.

So, suppose $A = LU$ and we want to solve

$$A\mathbf{x} = \mathbf{b} \tag{1.2}$$

for a vector $\mathbf{b}$. Substitute,

$$LU\mathbf{x} = \mathbf{b}.$$

Multiply by L^{-1}

$$L^{-1}LU\mathbf{x} = U\mathbf{x} = L^{-1}\mathbf{b}.$$

So, we have the system $U\mathbf{x} = L^{-1}\mathbf{b}$ and we can now multiply by U^{-1}

$$U^{-1}U\mathbf{x} = \mathbf{x} = U^{-1}L^{-1}\mathbf{b}.$$

This entirely oversimplifies this whole procedure. How do we compute L^{-1}, U^{-1}? And then we have to multiply by them? This is something that is accomplished all at once.

Theorem 1.21. *Suppose* $U \in \mathbb{R}^{n\times n}$ *is upper triangular with non-zero diagonal entries. Then computing* $U^{-1}\mathbf{c}$ *gives the exact same result as solving* $\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right]$ *with back substitution.*

This implies that (1) we never need to actually find out what U^{-1} is and (2) we already have an algorithm for handling this computation.

For this procedure to be successful we need to have an analogous procedure to handle L. Since L is lower-triangular, we can solve with forward substitution.

Algorithm 3: Forward substitution `Forsub`.

Result: The solution of $L\mathbf{c} = \mathbf{b}$
Input: An augmented matrix $\left[\begin{array}{c|c} L & \mathbf{b} \end{array}\right]$, $L \in \mathbb{R}^{n\times n}$ lower triangular and $\ell_{jj} = 1$ for all j
begin
  set $c_1 = b_1$;
  for $i = 2$ *to* n **do**
    set $c_i = b_i - \sum_{j=1}^{i-1} \ell_{ij} c_j$;
  end
end

Theorem 1.22. *Suppose* $L \in \mathbb{R}^{n\times n}$ *is lower triangular with ones on the diagonal. Then computing* $L^{-1}\mathbf{b}$ *gives the exact same result as solving* $\left[\begin{array}{c|c} L & \mathbf{b} \end{array}\right]$ *with forward substitution.*

With this in hand we can now describe the use of the LU factorization — if we have it.

Algorithm 4: Solve a system using an LU factorization

Result: The solution of $LU\mathbf{x} = \mathbf{b}$
Input: $U \in \mathbb{R}^{n\times n}$ upper triangular and $u_{jj} \neq 0$ for all j and a special lower-triangular matrix $L \in \mathbb{R}^{n\times n}$
begin
  set $\mathbf{c} = \texttt{Forsub}(\left[\begin{array}{c|c} L & \mathbf{b} \end{array}\right])$;
  set $\mathbf{x} = \texttt{Backsub}(\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right])$;
end

Before describing how to compute the LU factorization, we pause to note that when we applied regular Gaussian elimination to the augmented system $\left[\begin{array}{c|c} A & \mathbf{b} \end{array}\right]$ and reduced it to $\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right]$, the forward substitution step is not needed (indeed, it is actually encoded within regular Gaussian elimination!).

Algorithm 5: Solve a system using regular Gaussian elimination

---

Result: The solution of $A\mathbf{x} = \mathbf{b}$
Input: $A \in \mathbb{R}^{n\times n}$, regular.
begin
  set $\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right] = \texttt{rGE}(\left[\begin{array}{c|c} A & \mathbf{b} \end{array}\right])$;
  set $\mathbf{x} = \texttt{Backsub}(\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right])$;
end

1.5 ▪ Computing the LU factorization

Part II

Approximation theory

Chapter 2

Interpolation

To come!

Bibliography